SATViz:
Real-Time Visualization of Clausal Proofs

Tim Holzenkamp, Johann Zuber

August 1^{st} 2022

Karlsruhe Institute of Technology: KIT

Introduction

SATViz

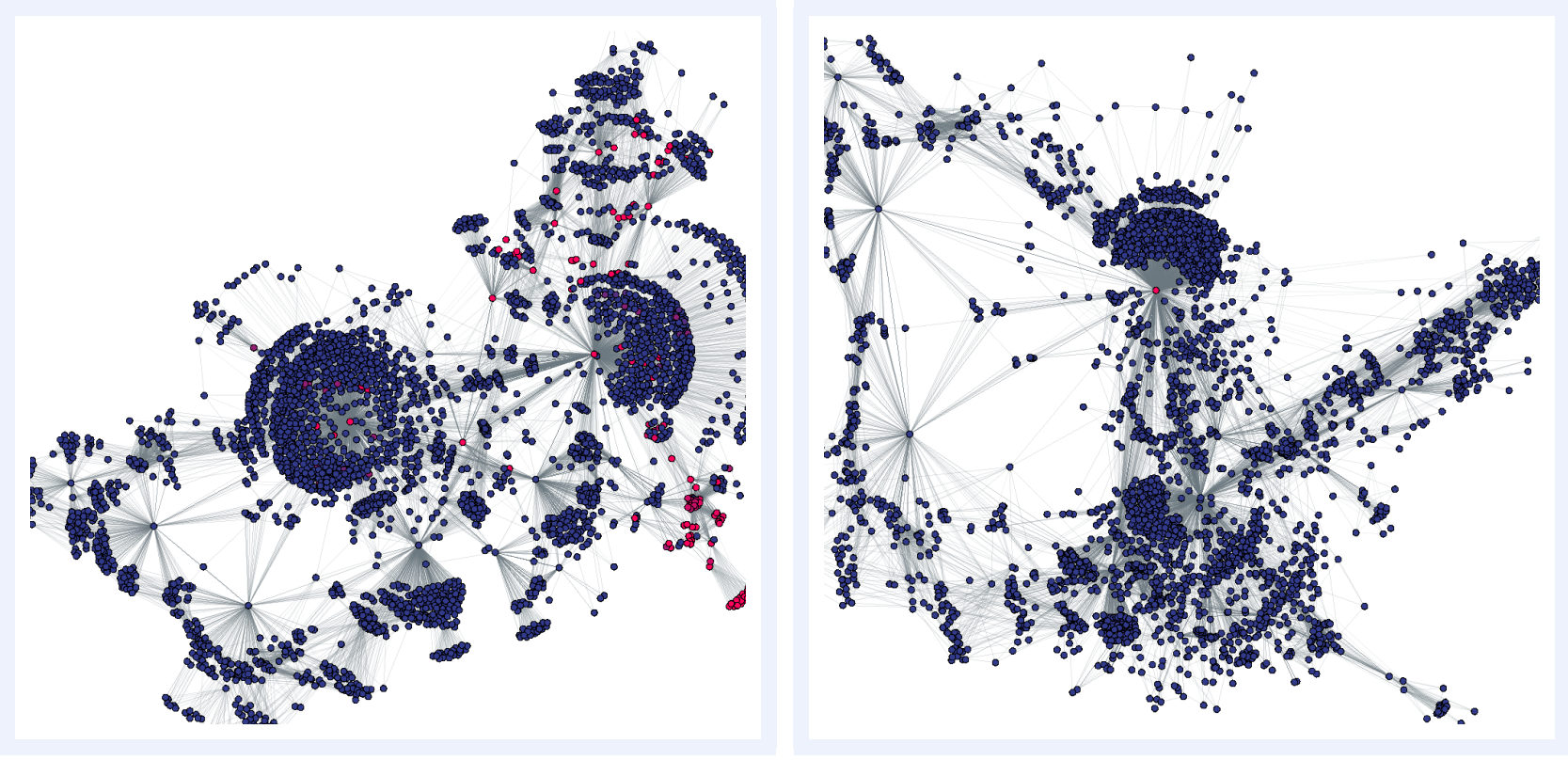


Figure 1: visualization of NEWTON.5.1.I.SMT2-CVC solved by KISSAT

Project Context

- Semester project built by students
- No prior background in SAT research
- Idea, requirements and guidance came from our supervisors

Motivation

- Visualize Conflict-Driven Clause Learning to
  - Understand the structure of SAT instances
  - Analyze the behavior of CDCL-solvers
- Extensible framework for processing clauses
- Interactive user interface
- Real-time changes
- Support for standard formats

System Architecture

Architecture Overview

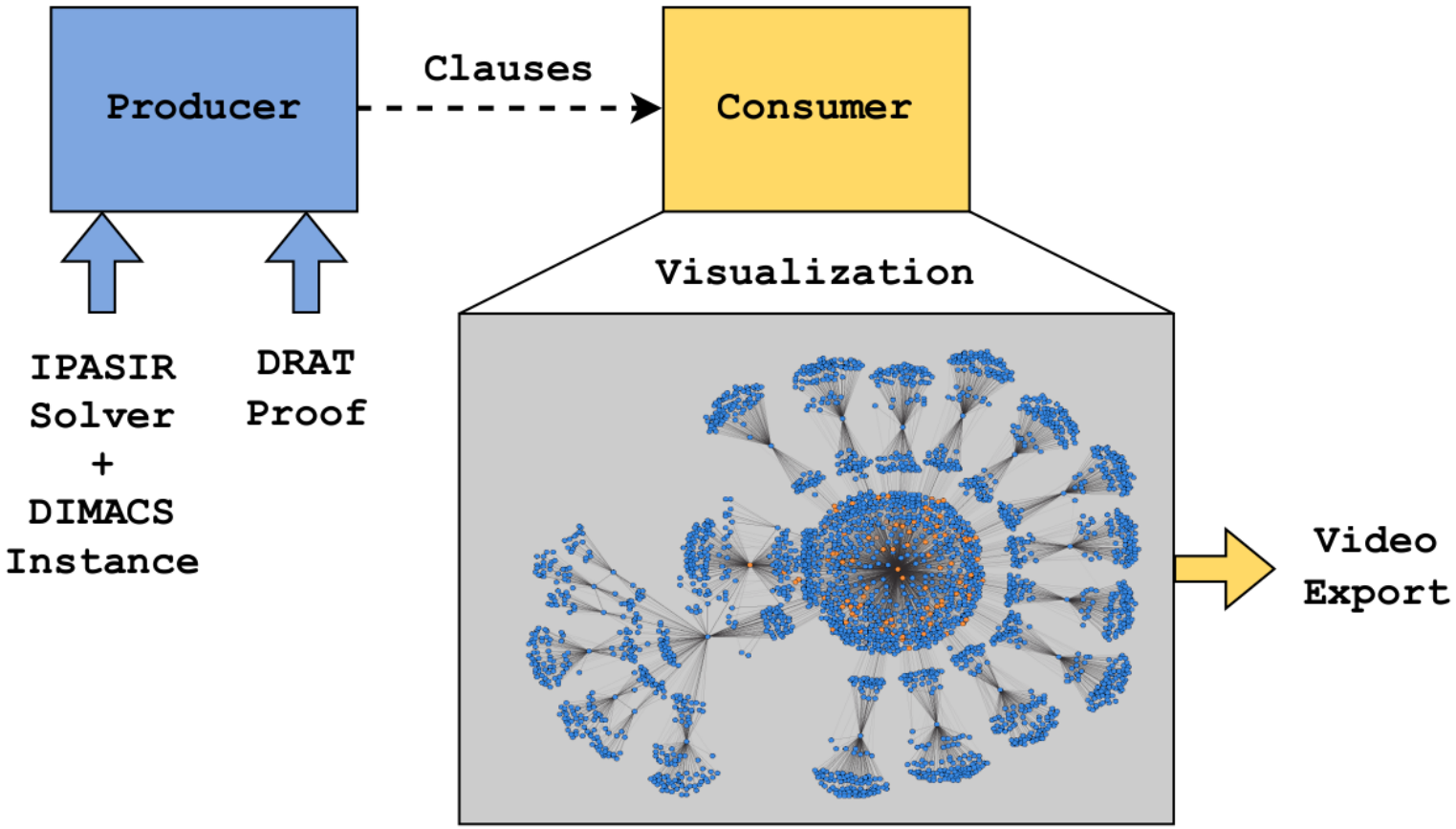
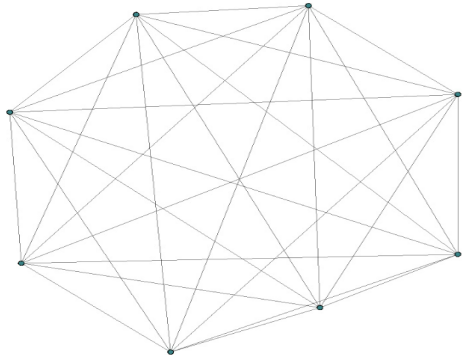


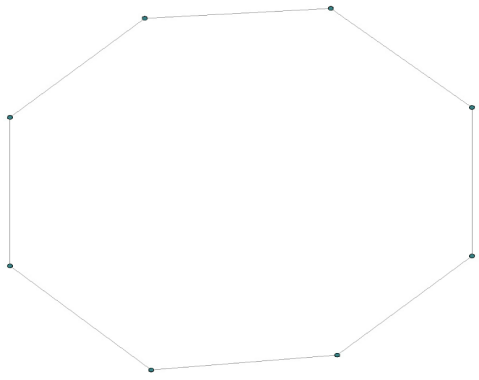
Figure 2: Overview of satviz Architecture

Algorithms

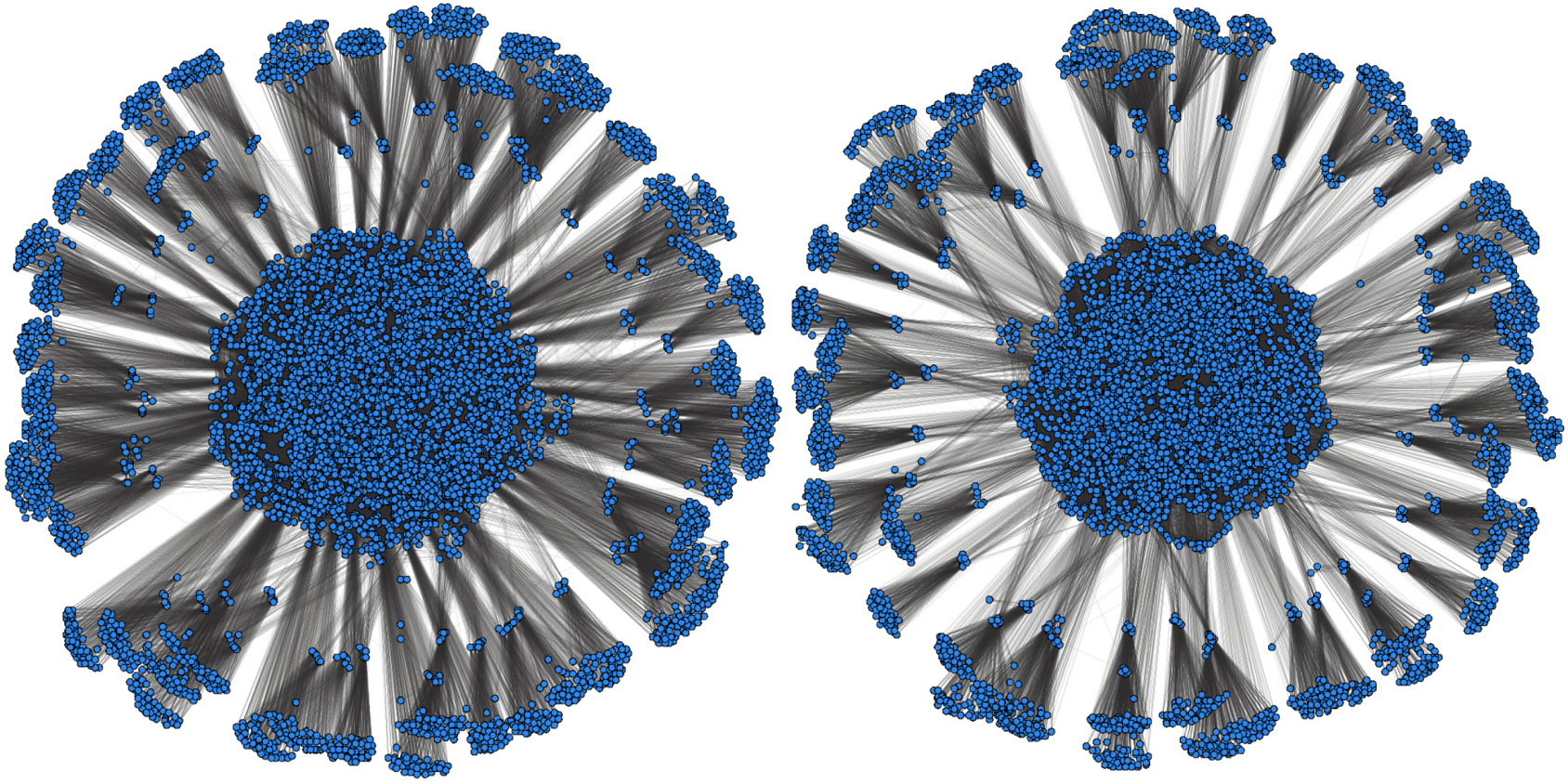
Visualization

- Undirected graph; 1 node $\leftrightarrow$ n variables
- 2 components to calculate edge weights and node colors:
  - *Variable Interaction Graph*
    (clique-expansion of CNF-hypergraph)
  - *Heatmap* (relative frequencies of variables)
- Original plans not entirely practical
- **Performance**: clique embedding $\Rightarrow$ ring embedding
- **Visual gain**: frequency-heat $\Rightarrow$ recency-heat





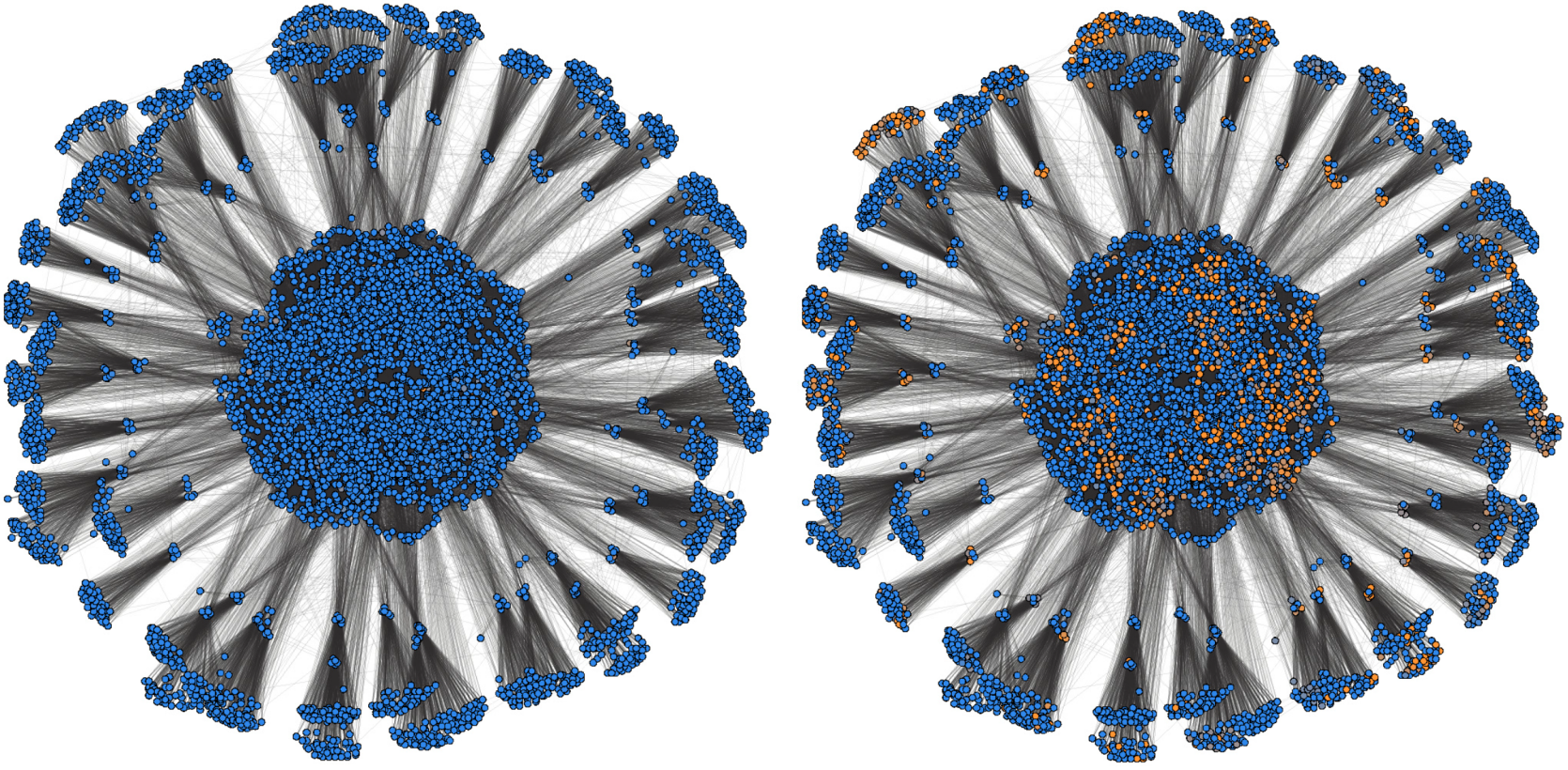
Variable Interaction Graph



(a) clique embedding

(b) ring embedding

Heatmap



(a) frequency-based heatmap

(b) recency-based heatmap

Graph Contraction

- Many SAT instances are **big** (> 50.000 variables)
  - Performance struggles
  - Graph becomes unintelligible

$\rightarrow$ **Reduce** the graph

  - Simple, experimental algorithm

Our contraction algorithm

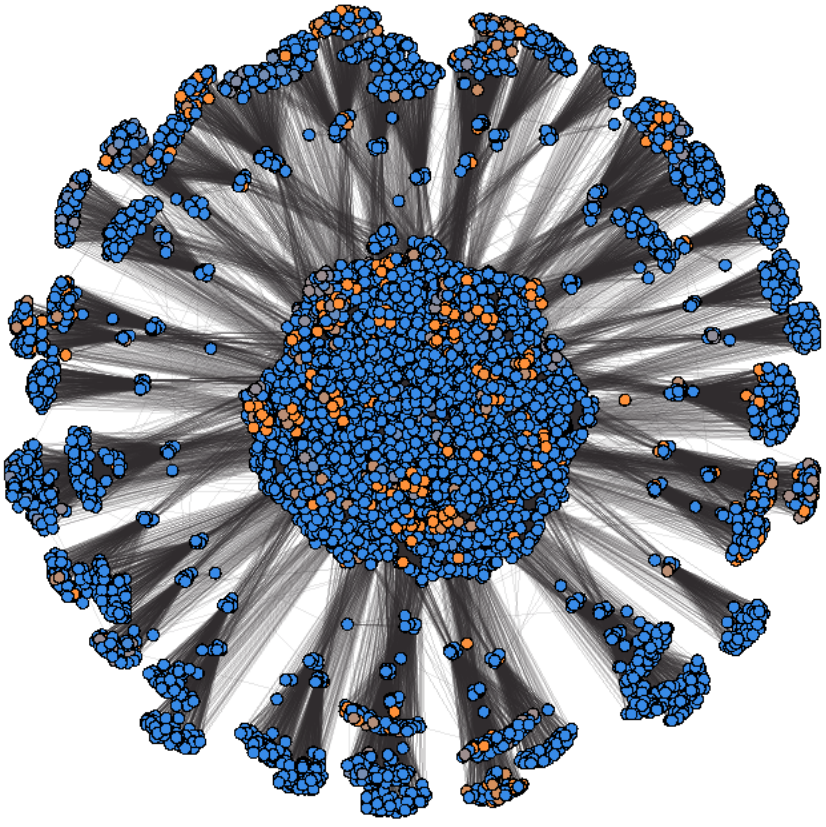
Our original algorithm:

- For each node, find strongest incident edge
- Collapse all found edges *
- Repeat N times

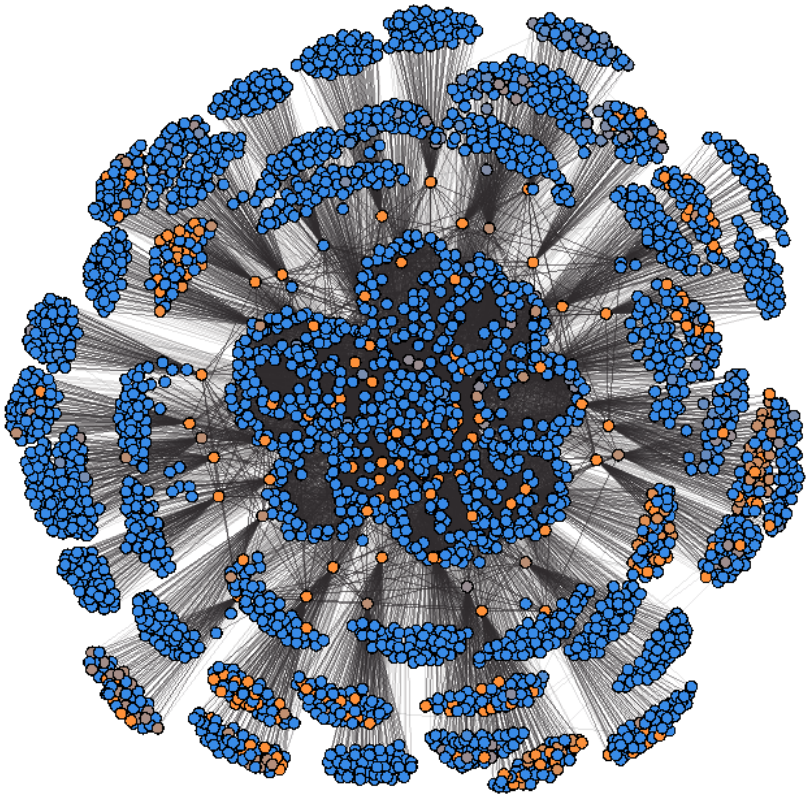
Improvement:

- * Only collapse strongest 30%

Graph Contraction



(a) 0 iterations (no contraction)



(b) 2 iterations

Contraction succeeded in preserving overall structure

Features

Features

- A lot of configuration options
  - Video recording settings
  - Visualization algorithm parameters
  - Contraction iterations
  - Embedded/external clause source
  - ...
- Many options also configurable at runtime

Features

- Multiple user interfaces: GUI or CLI
  - GUI offers interactivity and approachable settings
  - CLI is useful for power users, scripting
- Clause producer and visualisation can be run separately, across different machines
- Support for:
  - IPASIR shared libraries
  - DRAT proofs
  - DIMACS CNF instances
  - xz compression
  - Theora (`.ogv`) videos

Conclusion

Future Work

- Incremental graph layout algorithm (out of scope)
- Support for variable equivalence classes
- A producer implementation for distributed solving
- Refined graph contraction algorithm
- Different-sized nodes based on impact of contraction
- Improved CLI (support all parameters)
- Bug fixes

Getting your hands on SATViz

The project is open source (MIT License) and can be found at `https://github.com/satviz`

- Installation instructions in `README.md`
- Code base: mostly Java; Graphics and graph model in C++

System requirements:

- GNU/Linux on PC hardware
- Support for OpenGL 3.3 or newer
- Decent hardware (especially CPU, memory)

Thank you for your attention!

Any questions?